P systems and unique-sum sets

Pierluigi Frisco
School of Math. and Comp. Sciences,
Heriot-Watt University,
EH14 4AS Edinburgh, UK,
P.Frisco@hw.ac.uk

Abstract. We study P systems with symport/antiport and a new model of purely catalytic P systems, called *purely multi-catalytic P systems*, when these devices use only one symbol. Our proofs use unique-sum sets, sets of integer numbers whose sum can only be obtained in a unique way with the elements of the set itself.
We improve some results related to the descriptional complexity of the P systems with symport/antiport considered by us and we define one infinite hierarchy of computations.

1 Introduction

Membrane systems (*P systems*) are an abstract model of computation inspired by the compartmentalisation present in eukariotic cells [15, 16, 6, 18]. For several models of P systems it is possible to define rewriting systems without any compartment able to simulate (that is, mimic the behaviour of) the original P system. This is possible through the use in the rewriting systems of several symbols encoding the compartments of the P system and the passage of the symbols in the P system from one compartment to another.

This possibility to remove the compartments from a P system is something well known in this field of research and it has been formally studied through the use of Petri nets [6, 19].

Some models of P systems do not allow a reduction to rewriting systems. If, for instance, a P system with several compartments is ought to have only one symbol, then no other symbol can be introduced in a rewriting system simulating the P system. So, it makes a lot of sense to study P systems with restrictions in the number of used symbols, as these systems are the ones that do need the compartments in order to operate.

The computational power of *P systems with symport/antiport*, one of the most elegant and the most studied model of P system [14], has been studied when restrictions on the number of used symbols are in place [17, 1, 2, 6, 11].

Here we continue this line of research on P systems with symport/antiport proving that when these devices operate under maximal strategy, then the model using only one symbol and with $2n+3, n \geq 2$, compartments can simulate register machines. In this way we partially answer suggestion for research 5.3 in [6] and a problem stated in [11]. We also prove that when the number of occurrences of the unique symbol is bounded, then an infinite hierarchy on the number of compartments is present. We also study *purely multi-catalytic P systems* a model

of *catalytic P systems* introduced in the present paper, when these systems have several catalysts and only one symbol which is not a catalyst.

The provided proofs employ the use of *unique-sum sets*: sets of integer numbers whose sum can only be obtained in a unique way with the elements of the set itself.

2 Basic definitions

We assume the reader to have familiarity with basic concepts of formal language theory [9], register machines and P systems [6, 18]. In this section we recall particular aspects relevant to our presentation.

The symbol $\mathbb{N}$ denotes the set of natural numbers $\{0, 1, 2, \ldots\}$, while $\mathbb{N}_+$ denotes $\{1, 2, 3, \ldots\}$ and $\emptyset$ denotes the empty set.

Given a set V its *cardinality*, that is, the number of elements in V, is denoted with $|V|$; V^* denotes the free monoid generated by V with the operation of concatenation, λ indicates the empty word.

Let a grammar $G = (N, T, S, P)$ be of type-0 or of type-3. The length of the strings over T which can be obtained by derivations of G is the *set of numbers generated* by G. The respective classes of numbers are denoted by $\mathbb{N}\,\mathsf{REG}$ and $\mathbb{N}\,\mathsf{RE}$.

For $k \in \mathbb{N}$, $\mathbb{N}_k\,\mathsf{RE}$ equals the family of recursively enumerable sets with elements greater than or equal to k i.e., $\{L \in \mathbb{N}\,\mathsf{RE} \mid \{0, \ldots, k-1\} \cap L = \emptyset\}$, or equivalently, $\{k+L \mid L \in \mathbb{N}\,\mathsf{RE}\}$, where $k+L = \{k+n \mid n \in L\}$. From the point of computational completeness, the families $\mathbb{N}\,\mathsf{RE}$ and $\mathbb{N}_k\,\mathsf{RE}$ are equivalent, as a Turing machine can make the translation, but here we inherit the language definition used in the literature of P systems and make this distinction.

A *multiset* (over V) is a function $M : V \to \mathbb{N} \cup \{+\infty\}$; for $a \in V$, $M(a)$ defines the *multiplicity* of a in the multiset M. We will say that an element a of a multiset M has *infinite multiplicity* if $M(a) = +\infty$. In case the multiplicity of an element of a multiset is 1 we will indicate just the element, otherwise $(a, M(a))$ is indicated. The *support* of a multiset M is the set $supp(M) = \{a \in V \mid M(a) > 0\}$. The *size* of a multiset is defined by the function $|\ | : (V \to \mathbb{N} \cup +\{\infty\}) \to \mathbb{N} \cup \{+\infty\}$, where for M multiset over V, $|M| = \sum_{a \in supp(M)} M(a)$. The symbol ϕ indicates the *empty multiset*, that is the multiset whose support is the empty set.

Let $M_1, M_2 : V \to \mathbb{N} \cup \{+\infty\}$ be two multisets. The *union* of M_1 and M_2 is the multiset $M_1 \cup M_2 : V \to \mathbb{N} \cup +\{\infty\}$ defined by $(M_1 \cup M_2)(a) = M_1(a) + M_2(a)$, for all $a \in V$. The *difference* $M_1 - M_2$ is here defined only when M_2 is included in M_1 (which means that $M_1(a) \geq M_2(a)$ for all $a \in V$) and it is the multiset $M_1 - M_2 : V \to \mathbb{N} \cup \{+\infty\}$ given by $(M_1 - M_2)(a) = M_1(a) - M_2(a)$ for all $a \in V$. Of course, if $M_1(a) = +\infty$ and $M_2(a)$ is finite, then $M_1(a) - M_2(a) = +\infty$.

In this paper we describe formal systems simulating other formal systems. The concept of simulation is widely used in (theoretical) computer science and it refers to the fact that a device can mimic the behaviour of another device. In the present paper we use the definition of this concept as given in [7].

2.1 P systems

In this section we introduce the models of P system used in the present paper. Before doing so we define *cell-trees*.

A directed graph $\mu = (N, E)$, where N is the finite set of *vertices* and E is the set of *edges*, is said to be a *cell-tree* if:

- $0 \in N$ where the vertex 0 is the *root* of the tree;
- each vertex in N except the root defines a *membrane compartment* (in this paper referred simply as *compartment*) of a P system Π; the root 0 defines the *environment*;
- the root has only one child; this last is called *skin compartment*;
- the set E is the 'father of' relation present in μ equivalent to the nesting of membranes normally used in the literature of P systems.

In the following we represent cell-tree as boxes with a subscript, the number of the vertex represented by them, and eventually containing other boxes and symbols.

An *accepting P system with symport/antiport* [14, 6, 18] of *degree* m, $m \geq 1$, is a construct

$$\Pi = (V, \mu, L_0, L_1, \ldots, L_m, R_1, \ldots, R_m, comp)$$

where:

- V is a finite set of symbols;
- $\mu = (N, E)$ is a cell-tree with m vertices underlying Π;
- $L_i, 0 \leq i \leq m$, are multisets over V defining the initial multisets of symbols. All the symbols in L_0 have infinite multiplicity while the ones in $L_1, \ldots, L_m$ do not;
- $R_i, 1 \leq i \leq m$, are sets containing a finite number of *rules* of the form: $(v; \text{in}), (v; \text{out})$ (called *symport* rules), or $(w; \text{out}/v; \text{in})$ (called *antiport* rules), with v, w nonempty multisets over V with a finite support. Thus the symports $(\phi; \text{in})$ and $(\phi; \text{out})$ and the antiports $(b; \text{out}/a; \text{in})$ with $a = \phi$ or $b = \phi$ are not allowed;
- $comp \subset \{1, \ldots, m\}$ is the set of *initial compartments*. It is common practice to have $|comp| = 1$, anyhow, in this paper we will consider also cases in which $|comp| > 1$.

A *configuration* of a P system with symport/antiport of degree m is given by the $m + 1$-tuple $(M_0 - L_0, M_1, \ldots, M_m)$ of multisets over V associated to the environment and the compartments $\{1, \ldots, m\}$ respectively. Note that the

configuration does not record the symbols in the environment that occur with infinite multiplicity as they are invariant to any configuration. The $m+1$-tuple $(\phi, L_1, \ldots, L_m)$ is called *initial configuration*. For two configurations $(M_0 - L_0, M_1, \ldots, M_m), (M'_0 - L_0, M'_1, \ldots, M'_m)$ of Π we write $(M_0 - L_0, M_1, \ldots, M_m) \Rightarrow (M'_0 - L_0, M'_1, \ldots, M'_m)$ indicating a *transition* from $(M_0 - L_0, M_1, \ldots, M_m)$ to $(M'_0 - L_0, M'_1, \ldots, M'_m)$ that is the application of a set of rules associated to each compartment under the requirement of *maximal strategy*: in such a multiset a rule is applied at most once and all the rules that can be applied are actually applied (an example follows). If more than one maximal set of rules can be applied, then exactly one of them is nondeterministically chosen; all rules present in this set are applied in parallel.

The rules R_q associated to a compartment $q \in N \backslash \{0\}$ can change the multisets M_q and M_p of p father of q in μ in the following way:

- a multiset v included in M_p may be subtracted from M_p and may be united to M_q, if the symport rule $(v; \text{in})$ is present in R_p. In this case the multisets change from M_p and M_q to $M'_p = M_p - v$ and $M'_q = M_q \cup v$ respectively;
- a multiset v included in M_q may be subtracted from M_q and united to M_p if the symport rule $(v; \text{out})$ is present in R_q. The multisets change from M_p and M_q to $M'_p = M_p \cup v$ and $M'_q = M_q - v$ respectively;
- a multiset v included in M_p may be united to M_q while, at the same time, a multiset w included in M_q may be united to M_p if the antiport rule $(w; \text{out}/v; \text{in})$ is present in R_q. In this case the multisets of symbols change from M_p and M_q to $M'_p = (M_p - v) \cup w$ and $M'_q = (M_q - w) \cup v$ respectively.

In general, if a multiset v is subtracted from M_p and united to M_q we will say that v *passes* from compartment p to compartment q.

The *weight* of a rule is given by $|v|$ (that is, the cardinality of the multiset v) in case of a symport $(v; \text{in})$ or $(v; \text{out})$ and by $max(\{|v|, |w|\})$ in case of an antiport $(v; \text{out}/w; \text{in})$.

A *computation* is a finite sequence of transitions between configurations of a system Π starting from the initial configuration $(\phi, L_1, \ldots, L_m)$. If a computation is finite, then the last configuration is called *final* and we say that the system *halts*.

The result of the computation is given by the vector of multiset of symbols (the vector has dimension $|comp|$, if this dimension is 1, then we speak of set of numbers) present in the compartments in *comp* in the initial configuration when, on such initial configuration, the P system halts (that is, when the multiset of applicable rules has empty support).

The set of numbers whose elements are bigger than $k, k \in \mathbb{N}_+$, accepted by P systems with symport/antiport operating under maximal strategy using s symbols, degree at most m, using symports of weight at most p and antiports of weight at most q is denoted by $\mathbb{N}_k \mathsf{aO}_s \mathsf{P}_m(\text{sym}_p, \text{anti}_q)_{ms}$. When p, q or m are not bounded, then they are replaced by $*$. In this paper we prefer to use the subscript ms when we denote sets of numbers accepted by formal systems, because we want to emphasise that these sets are obtained by systems operating under maximal strategy.

The following example aims to clarify how maximal strategy operates. Let $\Pi = (\{a, b\}, \mu, \phi, \{a^3, b^2\}, \phi, \emptyset, R_2, \{2\})$ be a P system with symport/antiport operating under maximal strategy with:

$\mu = \boxed{\boxed{\ }_2}_1$;

$R_2 = \{1 : (a; \text{in}), 2 : (ab; \text{in}), 3 : (a^2b^2; \text{in}), 4 : (a^3b^2; \text{in})\}$.

In order to facilitate the explanation rules have been numbered. In the initial configuration of Π the applicable rules are: (1, 2), (1, 3) and 4 (where rule numbers in parenthesis denote that the rules are applied in parallel).

Differently than maximal parallelism, in maximal strategy in a configuration a rule can be applied at most once. This is present in the example, as, for instance, the triple application of rule 1 is not allowed even if possible. Moreover, maximal strategy does not aim to maximise the number of applied rules. In the example, either rules (1, 2), or rules (1, 3) or rule 4 can be applied.

It should be clear that when a P system operates under maximal strategy it makes sense to have rules in more than one copy. If, for instance, in previous example the set R_2 contained also $1' : (a; \text{in})$ and $1'' : (a; \text{in})$, then some of the applicable rules would be: $(1, 1', 1''), (1, 1', 2), (1'', 3)$, etc.

We now introduce a model of purely catalytic P systems [15, 6, 18], called *purely multi-catalytic P systems*, with limitations in its alphabet and having a new kind of rules.

An *accepting purely multi-catalytic P system* of *degree* m is a construct

$$\Pi = (V, C, \mu, L_1, \ldots, L_m, R_1, \ldots, R_m, comp)$$

where:

$V = \{a\}$ is an alphabet;
$C, C \cap V = \emptyset$, is a set of *catalysts*;
$\mu = (N, E)$ is a cell-tree with m vertices underlying Π;
$L_i, 1 \leq i \leq m$, are multisets over V defining the initial multisets of symbols;
$R_i, 1 \leq i \leq m$, are finite sets of *rules* of the kind: $ca^p \rightarrow c(a^q, tar)$ with $p, q \in \mathbb{N}_+$, $c \in C$ and $tar \in \{here, in, out\}$, where the set $\{here, in, out\}$ contains *target indicators*;
$comp \in \{1, \ldots, m\}$ is the *initial compartment*.

A *configuration* of Π is an m-tuple $(M_1, \ldots, M_m)$ of multisets over V associated with the compartments of Π. The m-tuple $(L_1, \ldots, L_m)$ is called *initial configuration*.

Let $\Pi = (V, C, \mu, L_1, \ldots, L_m, R_1, \ldots, R_m, comp)$ be an accepting purely multi-catalytic P system and let j, i and k be three vertices in μ such that i is the parent of j and j is the parent of k. Moreover, let R_j be the set of rules associated with compartment j and let M_j, M_i and M_k be multisets over V, associated with vertices j, i and k, respectively, such that $ca^p \rightarrow c(a^q, tar) \in R_j, p, q \in \mathbb{N}_+$ and $tar \in \{here, in, out\}$. Depending on the value of tar the application of $ca^p \rightarrow c(a^q, tar) \in R_j$ changes the multisets M_j, M_i and M_k according to the following:

if $tar = here$, then M_j becomes $M'_j = M_j \setminus \{a^p\} \cup \{a^q\}$ while M_i and M_k remain unchanged;
if $tar = in$, then M_j becomes $M'_j = M_j \setminus \{a^p\}$, M_k becomes $M'_k = M_k \cup \{a^q\}$ and M_i remains unchanged;
if $tar = out$, then M_j becomes $M'_j = M_j \setminus \{a^p\}$, M_i becomes $M'_i = M_i \cup \{a^q\}$ and M_k remains unchanged.

In the following the target indicator *here* is omitted. For two configurations $(M_1, \ldots, M_m)$ and $(M'_1, \ldots, M'_m)$ of Π we write $(M_1, \ldots, M_m) \Rightarrow (M'_1, \ldots, M'_m)$ to denote a *transition* from $(M_1, \ldots, M_m)$ to $(M'_1, \ldots, M'_m)$, that is, the application of a multiset of rules associated with each compartment under the requirement of maximal strategy.

A *computation* is a sequence of transitions between configurations of a system Π starting from the initial configuration $(L_1, \ldots, L_m)$. If a computation is finite, then the last configuration is called *final* and we say that the system *halts*.

Given a purely multi-catalytic P system $\Pi = (V, C, \mu, L_1, \ldots, L_m, R_1, \ldots, R_m, comp)$ with $comp = c_1$, we consider two results from a computation of Π:

$\mathsf{N}(\Pi)$: the sum of occurrences of a and catalysts present in *comp* in the initial configuration, when, on such initial configuration, Π halts;
$\mathsf{N}_{-c}(\Pi)$: the occurrences of a present in *comp* in the initial configuration, when, on such initial configuration, Π halts.

Moreover,

$\mathbb{N}_k\mathsf{aO}_1\mathsf{P}_m(\mathrm{pmcat}_p)_{ms} = \{\mathsf{N}(\Pi) \mid \Pi$ is an accepting purely multi-catalytic P system operating under maximal strategy with degree at most m and using at most p catalysts and such that each element in $\mathsf{N}(\Pi)$ is bigger than $k, k \in \mathbb{N}_+\}$;
$\mathbb{N}\mathsf{aO}_1\mathsf{P}_{m,-c}(\mathrm{pmcat}_p)_{ms} = \{\mathsf{N}_{-c}(\Pi) \mid \Pi$ is an accepting purely multi-catalytic P system operating under maximal strategy, of degree at most m and using at most p catalysts$\}$;

So, for instance, if such a P system has $comp = 2$ and the only initial configuration for which the P system halts is such that compartment 2 has two catalysts and three occurrences of a, then:

$\{5\}$ is the set accepted if all symbols counted with their multiplicity are considered;
$\{3\}$ is the set accepted if only the a symbols are considered;

2.2 Register machines

A *register machine* [13] (also known as *(multi)counter machines*, *multipushdown machines*, *program machine* and *counter automata*) with n registers ($n \in \mathbb{N}_+$) is a construct $M = (S, R, s_1, s_f)$, where:

S is a finite set of *states*;

R is a finite set of *instructions* of the form $(s_p, \gamma_t^-, s_q), (s_p, \gamma_t^+, s_q)$ or $(s_p, \gamma_j{}^{=0}, s_q)$, with $s_p, s_q \in S,\ s_p \neq s_f, 1 \leq t \leq n$;
$s_1, s_f \in S$ are respectively called the *initial* and *final* states.

A *configuration* of a register machines M with n registers is given by an element in the $n+1$-tuples $(s,\ \mathbb{N}^n),\ s \in S$. Given two configurations $(s, val(\gamma_1), \ldots, val(\gamma_n))$ and $(s', \gamma_1', \ldots, \gamma_n')$ (where $val : \{\gamma_1, \ldots, \gamma_n\} \to \mathbb{N}$ is the function returning the content of a register) we define a *computational step* as $(s, val(\gamma_1), \ldots, val(\gamma_n)) \vdash (s', \gamma_1', \ldots, \gamma_n')$:

- if $(s, \gamma_t^-, s_q) \in S$ and $val(\gamma_t) \neq 0$, then $s' = s_q,\ \gamma_t' = val(\gamma_t) - 1,\ \gamma_k' = val(\gamma_k),\ k \neq t,\ 1 \leq k \leq n$;
  if $val(\gamma_t) = 0$, then the register machine halts in the non-final state s;
  (informally: in state s if the content of register γ_t is greater than 0, then subtract 1 from that register and change state into s_q, otherwise halt in a non-final state);
- if $(s_p, \gamma_j^{=0}, s_q) \in S$ and $val(\gamma_t) = 0$, then $s' = s_q,\ \gamma_k' = val(\gamma_k),\ 1 \leq k \leq n$;
  if $val(\gamma_t) \neq 0$, then the register machine halts in the non-final state s;
  (informally: in state s if the content of register γ_t is 0, then change state into s_q, otherwise halt in a non-final state);
- if $(s, \gamma_t^+, s_q) \in S$, then $s' = s_q, \gamma_t' = val(\gamma_t) + 1,\ \gamma_k' = val(\gamma_k),\ k \neq t,\ 1 \leq k \leq n$;
  (informally: in state s add 1 to register γ_t and change state into s_q).

A *computation* is a sequence of computational steps of a register machine M starting from the *initial configuration* $(s_1, val(\gamma_1), 0, \ldots, 0)$. If a computation is finite, then the last configuration is called *final*. If a final configuration has s_f as state, then we say that M *halts* and it *accepts* the input $val(\gamma_1)$. For this reason γ_1 is called the *input register* and M is called an *accepting register machine*. Starting from an initial configuration $(s_1, val(\gamma_1), 0, \ldots, 0)$ a register machine M could have a finite sequence of computational steps in which the last one does not have s_f as state. In this case we say that M *stops* and $val(\gamma_1)$ is not accepted.

Partially blind register machines [8] are defined as register machines without test on zero. The only allowed operations are (s, γ^+, s') and (s, γ^-, s') where γ is a register. In case the machine tries to subtract from a register having value zero it halts in a non-final state. Partially blind register machines are strictly less powerful from a computational point of view than register machines.

We summarise in the following proposition, results from [13, 3]:

Proposition 1 *Register machines with three registers can accept* $\mathbb{N}$ RE*, the number of registers can be decreased to two if specific input format (for example,* 2^x *instead of* x*) is used.*
Register machines with only one register can accept $\mathbb{N}$ REG.

Restricted register machines are defined as register machines restricted in their operations: they can increase the value of a register, say β, only if they decrease the value of another register, say γ, at the same time.

So, restricted register machines have only one kind of instruction: $(s, \gamma^-, \beta^+, v, w)$ with s, v, w states and γ, β different registers of the restricted register machine. If when in state s the content of register γ can be decreased by 1, then the one of register β is increased by 1 and the machine goes into state v, otherwise no operation is performed on the registers and the machine goes into state w.

Here a theorem from [10] that we will need:

Theorem 1. *Restricted register machines with n+1 registers are more powerful from a computational point of view than those with n registers.*

A consequence of this theorem is that an infinite hierarchy is induced, by means of the number of registers, among families of computed vectors of numbers.

2.3 Operational modes

Given a formal system, it is normally possible to let it operate (run) in different ways.

P systems have their operational mode embedded in their definition. This means that, for instance, a P systems Π with symport/antiport operating under maximal strategy operates *only* under maximal strategy. If one wants to run the same system Π in a different way, then it has to define it.

In this paper we follow this custom. Differently than the vast majority of P systems, the models of P systems considered in this paper do not operate under maximal parallelism but under maximal strategy. The term *maximal strategy* is borrowed from the Petri net nomenclature. In the literature of P systems maximal strategy has been defined under different names: in [12], for instance, it has been called k-Max Parallel (where k is the number of rules of the P system). Here we prefer to use the term *maximal strategy* because it was introduced earlier in time.

Studies of different operational models for P systems can be found in [6, 4].

2.4 Unique-sum sets

Some proof in this paper use the following mathematical concepts.

Definition 1 *Let $U = \{u_1, \ldots, u_p\}$ be a set of distinct natural numbers and $\sigma_U = \sum_{i=1}^{p} u_i$ the sum of the elements of U. The set U is said to be a unique-sum set if the equation $\sum_{i=1}^{p} c_i u_i = \sigma_U, c_i \in \mathbb{N}$, has only the solutions $c_i = 1$, $1 \leq i \leq p$.*

An example of unique-sum set is $U' = \{4, 6, 7\}$ as $4 + 6 + 7 = 17$ and 17 cannot be obtained with any other linear combination of 4, 6 and 7. The set $U'' = \{4, 5, 6\}$ is not a unique-sum set as $4 + 5 + 6 = 15 = 5 + 5 + 5$.

It should be clear that any subset of a unique-sum set is a unique-sum set, too. In particular none of the elements of a unique-sum set can be obtained as a linear combination of the remaining elements in the set.

Proposition 2 *The sets* $U_p = \cup_{m=1}^{p}\{2^p - 2^{p-m}\}$ *with* $p \in \mathbb{N}_+$, *are unique-sum sets.*

The sum of the elements of the sets in this family is $\sigma_{U_p} = (p-1)2^p + 1$. The first sets in this family are:

$$U_1 = \{1\};$$
$$U_2 = \{2, 3\};$$
$$U_3 = \{4, 6, 7\};$$
$$U_4 = \{8, 12, 14, 15\};$$
$$U_5 = \{16, 24, 28, 30, 31\};$$
$$U_6 = \{32, 48, 56, 60, 62, 63\}.$$

From [5] it is known that:

Theorem 2. *The family of sets indicated in Proposition 2 contains the unique-sum sets having minimal sum in function of their number of elements.*

3 P systems with symport/antiport

In [11] it was proved that P system with symport/antiport operating under maximal parallelism, with only one symbol and degree $2n + 3$ can simulate a partially blind register machines with n registers. In [11] it was also proved that if priorities are added to the rules, then the obtained P system, having $n+3$ compartments, can simulate register machines with n registers. The former result was improved in [6] where it was proved that any partially blind register machine with n registers can be simulated by a P system with symport/antiport with only one symbol, degree $n + 3$ and operating under maximal parallelism. Here we prove that P systems with symport/antiport operating under maximal strategy, with only one symbol and degree $2n+3$ can simulate register machines with n registers.

Theorem 3. *Any accepting register machines with n registers can be simulated by an accepting P system with symport/antiport operating under maximal strategy, using only one symbol and with degree* $2n + 3$.

Proof. Let $M = (S, I, s_1, s_f)$ be a register machine with n registers $\gamma_1, \ldots, \gamma_n$. We define the P system with symport/antiport operating under maximal strategy

$$\Pi = (\{a\}, \mu, \{a\}, L_1, \ldots, L_{n+2}, L_{2'}, \ldots, L_{n+2'}, R_1, \ldots, R_{n+2}, R_{2'}, \ldots, R_{n+2'}, \{2\})$$

where:

$$
\begin{aligned}
\mu &= \boxed{\boxed{\ }_{2}\ _{2'} \cdots \boxed{\ }_{n+1}\ _{n+1'}\ \boxed{\ }_{n+2}\ _{n+2'}}_{1};\\
L_1 &= a^{c(s_1)};\\
L_{j+1} &= a^{b(j)+2val(\gamma_j)},\ 1 \le j \le n;\\
L_{j+1'} &= a^{b(j)},\ 1 \le j \le n;\\
L_{n+2} &= a;\\
L_{n+3} &= a;\\
R_1 &= \{1 : (a^{c(p)}; \text{out}/a^{c^{(1)}(q)+b(t)+2}; \text{in}), 2 : (a^{c^{(1)}(q)}; \text{out}/a^{c^{(2)}(q)}; \text{in}),\\
&\quad 3 : (a^{b(t)+c^{(2)}(q)}; \text{out}/a^{c(q)}; \text{in}) \mid (s_p, \gamma_t^+, s_q) \in I\} \cup\\
&\quad \{4 : (a^{c(p)}; \text{out}/a^{b(t)+c^{(3)}(q)}; \text{in}), 6 : (a^{c(p)}; \text{out}/a^{c^{(3)}(q)+b(t)+2}; \text{in}),\\
&\quad 7 : (a^{c^{(3)}(q)}; \text{out}/a^{c^{(4)}(q)}; \text{in}),\\
&\quad 8 : (a^{c^{(4)}(q)+b(t)}; \text{out}/a^{c^{(5)}(q)+b(t)+1}; \text{in}),\\
&\quad 9 : (a^{c^{(5)}(q)}; \text{out}/a^{c^{(6)}(q)}; \text{in}), 11 : (a^{c^{(6)}(q)+b(t)}; \text{out}/a^{c^{(7)}(q)+b(t)}; \text{in}),\\
&\quad 13 : (a^{c^{(7)}(q)}; \text{out}/a^{c^{(8)}(q)}; \text{in}), 15 : (a^{c^{(8)}(q)+b(t)+5}; \text{out}/a^{c(q)}; \text{in})\\
&\qquad\qquad \mid (s_p, \gamma_t^-, s_q) \in I\} \cup\\
&\quad \{16 : (a^{c(p)}; \text{out}/a^{c^{(10)}(q)+5b(t)+2}; \text{in}), 17 : (a^{c^{(10)}(q)}; \text{out}/a^{c^{(11)}(q)}; \text{in}),\\
&\quad 21 : (a^{c^{(11)}(q)+b(t)}; \text{out}/a^{c^{(12)}(q)+b(t)+2}; \text{in}),\\
&\quad 22 : (a^{c^{(12)}(q)}; \text{out}/a^{c^{(13)}(q)}; \text{in}),\\
&\quad 25 : (a^{c^{(13)}(q)+5b(t)}; \text{out}/a^{c^{(14)}(q)+b(t)+1}; \text{in}),\\
&\quad 26 : (a^{c^{(14)}(q)}; \text{out}/a^{c^{(15)}(q)}; \text{in}),\\
&\quad 27 : (a^{c^{(15)}(q)+b(t)}; \text{out}/a^{c^{(16)}(q)+b(t)}; \text{in}),\\
&\quad 28 : (a^{c^{(16)}(q)}; \text{out}/a^{c^{(17)}(q)}; \text{in}),\\
&\quad 29 : (a^{c^{(17)}(q)+b(t)+5}; \text{out}/a^{c(q)}; \text{in}) \mid (s_p, \gamma_j^{=0}, s_q) \in I\} \cup\\
&\quad \{30 : (a^{c(s_f)}; \text{out})\}\\
R_{j+1'} &= \{3 : (a^{b(j)}; \text{out}/a^{b(j)+2}; \text{in}), 10 : (a^{b(j)}; \text{out}/a^{b(j)+1}; \text{in}),\\
&\quad 14 : (a^{b(j)+5}; \text{out}/a^{b(j)}; \text{in}), 18 : (a^{b(j)}; \text{out}/a^{5b(j)+2}; \text{in}),\\
&\quad 23 : (a^{5b(j)}; \text{out}/a^{b(j)+2}; \text{in})\} \qquad 1 \le j \le n;\\
R_{j+1} &= \{5 : (a^{b(j)}; \text{out}/a^{b(j)+2}; \text{in}), 12 : (a^{b(j)+3}; \text{out}/a^{b(j)-1}; \text{in}),\\
&\quad 19 : (a; \text{out}/a^{2b(j)}; \text{in}), 20 : (a; \text{out}/a^{2b(j)}; \text{in})\} \qquad 1 \le j \le n;\\
R_{n+2} &= \{31 : (a; \text{out}/a; \text{in})\};\\
R_{n+2'} &= \{32 : (a; \text{out}/a; \text{in})\}.
\end{aligned}
$$

In order to facilitate the explanation rules have been numbered.

This proof requires the use of a unique-sum U with at least $n + 18|S| + 1$ elements. Different multiplicities of a, where the multiplicities are elements in U, are associated with each of the registers and instructions of M. This is performed by the function $b : \{1, \ldots, n\} \to U$ and the eighteen functions $c,\ c^{(1)}, \ldots, c^{(17)}$ all from S to U, injective and with disjoint values.

The exact definition of these functions is irrelevant for the proof, the only thing that is essential is that the different elements of U are at least 3 units apart from each other. The reason for this requirement is explained in the following.

The simulation performed by the P system Π is strongly based on the use of a unique-sum set and on the property that for such sets none of the elements

can be obtained as a linear combination of the remaining elements in the set. During the computation of Π, different occurrences of the symbol a are present in the skin compartment. Specific sequences of applied rules allow Π to simulate instructions of M. Other sequences of applied rules let Π to never halt.

The compartments $j+1, j+1'$, $1 \leq j \leq n$, are uniquely associated with registers in M. Each of these compartments contains at least $b(j)$ occurrences of a in the initial configuration. This number of occurrences of a represents 0 as content of the registers in M. The addition of 1 to register γ_j in M is performed adding two occurrence of a to compartment $j+1$. Conversely for the subtraction. The presence of just $a^{c(s)}$, $s \in S$, in the skin compartment indicates that Π simulates the register machine being in state s.

In the following, rules in parenthesis denote their parallel application.

The simulation of instructions of the kind (s_p, γ_t^+, s_q) is performed by the sequential application of rules 1, (2, 3), (4, 5).

The simulation of instructions of the kind (s_p, γ_t^-, s_q), if in compartment $t+1$ there are at least $b(t)+2$ occurrences of a, is performed first adding a^3 to compartment $t+1$, and then subtracting a^5 from the same compartment. This is performed by the sequential application of rules 6, (7, 3), (8, 5), (9, 10), (11, 12), (13, 14), 15. If in compartment $t+1$ there are less than $b(t)+2$ occurrences of a, then rule 12 cannot be applied. In this case, one occurrence of a brought in the skin compartment by rule 11 is used by rule 31. This starts the infinite application of rule 32, so that Π never halts.

The simulation of instructions of the kind $(s_p, \gamma_t^{=0}, s_q)$, if in compartment $t+1$ there are just $b(t)$ occurrences of a (simulating the counter t being empty), is performed by the sequential application of rules 16, (17, 18), (21, 5), (22, 23), (25, 5), (26, 10), (27, 12), (28, 14), 29.

Now we explain in more details this long sequence of applied rules. After the application of rules 6, (17, 18), $a^{5b(t)+2}$ is present in compartment $t+1'$. When this happens, several rules in R_{t+1} can be applied depending on the content of compartment $t+1$. It is important to notice that none of the rules in $R_{t+1'}$ can be applied without starting an infinite computation. The rules in these compartments are all antiports, that is, they need appropriate values in the skin compartment in order to perform simulations of instructions in M.

If compartment $t+1$ contains only $a^{b(t)}$, then either only rule 5 or (19, 20) are applied (rules 19 and 20 are equal, as we said in the example present in Section 2.1, this makes sense when maximal strategy is in use). If rules (19, 20) are applied, then the number of occurrences of a in compartment $t+1'$ becomes less than $5b(j)$. This means that rule 23 cannot be later applied, and Π starts an infinite computation (so, even if compartment $t+1$ contains only $a^{b(t)}$, the application of rules (19, 20) let the computation to never halt). If instead rule 5 is applied, then rule 23 can be later applied.

If compartment $t+1$ contains more than $a^{b(t)}$, then for sure at least rule 19 or 20 is applied (not allowing the later application of rule 23).

The application of rule 5 let 2 occurrences of a to be added to compartment $t+1$. These two occurrences are removed from compartment $t+1$ in the same

way the simulation of instructions of the kind (s_p, γ_t^-, s_q) takes place: first 2 occurrences are added to compartment $t+1$ and the 4 occurrences are remove from this compartment.

The reason why we ask that the elements of U are at least 3 units apart from each other is that because some configurations see a^2 present in the skin compartment. If the elements of U were not at least 3 units apart from each other there could be a rule using the number of occurrences of a's together with other a's present in the skin compartment. The application of this rule could let Π not to simulate M.

We stress that Π runs under maximal strategy (and not maximal parallelism). This means that in one configuration a rule can be applied at most once. This is essential in the system we presented (that is, Π would not simulate M if operating under maximal parallelism). For instance, if $a^{2b(j)+10}$ is present in compartment $t+1$, then rule 12 could be applied twice (under maximal parallelism) and this could lead to undesired behaviour.

In the above simulation many other sequences of configurations could occur but none of them would lead the system to a halt.

When $a^{c(s_f)}$ is present in compartment 1, then the application of rule 30 let $a^{c(s_f)}$ to pass to the environment and the computation halts.

The simulation of M is faithful, that is, Π cannot simulate sequences of instructions that cannot be performed by M. This means that if Π starts in a configuration in which $a^{c(s_1)}$ is present in the skin compartment, $a^{b(1)}$ is present in compartment $2'$, $a^{b(1)+2val(\gamma_1)}$ is present in compartment 2 and compartments $j+1, j+1', 3 \leq j \leq n$, contain $a^{b(j)}$, then Π halts with $a^{b(j)}$ in compartments $j+1, j+1', 1 \leq j \leq n$, only if M reached the final state with all registers empty if it started from the configuration $(s_1, val(\gamma_1), 0, \ldots, 0)$. □

From the previous theorem and Proposition 1 we have:

Corollary 1. *There exist $b, b', p, p', q, q' \in \mathbb{N}_+$ such that*

$\mathbb{N}_b \mathsf{O}_1 \mathsf{P}_9(sym_p, anti_q)_{ms} = \mathbb{N}_b \mathsf{RE}$;

$\mathbb{N}_{b'} \mathsf{O}_1 \mathsf{P}_7(sym_{p'}, anti_{q'})_{ms} = \mathbb{N}_{b'} \mathsf{RE}$ *if a specific input format is used,*

If a P system with symport/antiport Π is such that the skin compartment contains no rule then, for each computation of Π, the number and type of symbols in Π is constant and equal to the one of the initial configuration. In the following we consider such systems when they use only one symbol.

Theorem 4. *The set of vectors accepted by restricted register machines coincides with the ones accepted by P systems with symport/antiport operating under maximal strategy, using a constant number of occurrences of only one symbol.*

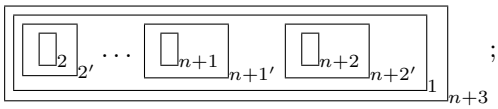
Proof. Part I: (These P systems can simulate restricted register machines) This proofs follows from the one of Theorem 3 after a few changes in the P system

there. Let $M = (S, I, s_1, s_f)$ be a restricted register machines with n registers $\gamma_1, \ldots, \gamma_n$.

We define the P system with symport/antiport operating under maximal strategy

$\Pi = (\{a\}, \mu, \{a\}, L_1, \ldots, L_{n+3}, L_{2'}, \ldots, L_{n+2'}, R_1, \ldots, R_{n+3}, R_{2'}, \ldots, R_{n+2'}, \{2, \ldots, n+1\})$

where:

μ = 

;

$L_{n+3} = a^\sigma$;

$R_{n+3} = \emptyset$;

the remaining elements of Π are defined as in the proof of Theorem 3 while σ is the sum of the unique sum set U used in that proof. Also the unique sum set U and the functions $b, c,\ c^{(1)}, \ldots, c^{(17)}$ are as defined in the proof of Theorem 3.

The fact that the number of occurrences of the only symbol is constant in Π derives from the fact that in restricted register machines the sum of the content of the counters is constant.

The simulation of the instructions $(s, \gamma^-, \beta^+, v, w)$ can be performed with the non-deterministic simulations of either $(s, \gamma^{=0}, w)$ or (s, γ^-, v') and (v', β^+, v), as explained in the proof of Theorem 3, where v' is a newly introduced state uniquely associated to $(s, \gamma^-, \beta^+, v, w)$.

It should be clear that, as the sum of the registers in M is constant, in Π occurrences of a needed to simulate the instructions of M are provided by compartment L_{n+3}. Moreover, if M starts in a configuration with $val(\gamma_1), \ldots, val(\gamma_n)$ in its registers, then Π starts in an initial configuration with $b(j) + 2var(\gamma_j)$ in compartment $j+1, 1 \leq j \leq n$. So, if M reaches its final state s_f on input $val(\gamma_1), \ldots, val(\gamma_n)$, then Π could halt on input $b(j) + 2var(\gamma_j), 1 \leq j \leq n$ accepting this vector of numbers.

If the simulated restricted register machine has n registers, then the simulating P system has $2n + 4$ compartments.

Part II: (Restricted register machines can simulate these P systems) Let Π be such a P system and let M be the restricted register machine simulating it. Each compartment of Π is associated to a different register in M whose content reflects the number of occurrences of the only symbol in the associated compartment. The simulations of each rule of Π is performed by a sequence of instructions decreasing and increasing the value of registers. With 'sequence of instructions' we mean instructions such that the output state of one instruction is the input state of the following instruction in the sequence.

Let, for instance, compartment s contain compartment r and let $(a^p; \text{out}/a^q; \text{in})$ belong to the set of rules associated to r. The set of instructions of M will then contain a sequence of p instructions decreasing the value of register r and increasing the one of register s followed by a sequence of q instructions decreasing the value of register s and increasing the one of register r. If the simulation of the whole rule cannot be performed, then M restores the content of the registers

(this can be done keeping track in the finite states of how far the simulation of a rule went) and M goes on trying to simulate another rule. The machine M tries to simulate all the rules in sequence and, if possible, a rule is simulated at most once in such a cycle (this is because Π works under maximal strategy). If in one cycle none of the rules could be simulated (again, the finite number of states can keep track of this), then M goes into its final state. □

We do not know if $2n+4$ is the minimum number of compartments needed for the P systems in Theorem 4 to simulate restricted register machines with n registers. Anyhow, as these P systems have a constant number of only one symbols, it is unlikely that such a P system with a number of compartment independent from n could simulate restricted register machines with an arbitrary number n of registers.

So, knowing Theorem 4 and that restricted register machines induce an infinite hierarchy on the number of registers, we feel confident to say that:

Corollary 2. *P systems with symport/antiport operating under maximal strategy, using a constant number of occurrences of only one symbol induce an infinite hierarchy on the number of compartments.*

4 Purely multi-catalytic P systems

In this section show how unique-sum sets can be used with purely catalytic P systems using only one symbol. It is important to notice that if a purely multi-catalytic P system has only 1 occurrence of each catalyst in each compartment (as in the systems considered in the present section), then maximal strategy and maximal parallelism coincide. Anyhow, in the following we only mention maximal strategy.

Theorem 5. *Any accepting register machines with n registers can be simulated by an accepting purely multi-catalytic P system operating under maximal strategy, using only one symbol and with degree $2n+3$.*

Proof. Let $M = (S, I, s_1, s_f)$ be an accepting register machine with n registers: $\gamma_1, \ldots, \gamma_n$. We define the purely multi-catalytic P system

$$\Pi = (\{a\}, C, \mu, L_2, R_1, \ldots, R_{2n+3}, 2)$$

where:

$C = \{c_1, c_\infty, c_{\infty'}\} \cup \{c_j, c_{j'}, c_{j'}^{(1)}, c_{j'}^{(2)} \mid 2 \le j \le n+1\}$;

$\mu = \left[\left[\right]_2 \right]_{2'} \cdots \left[\left[\right]_{n+1} \right]_{n+1'} \left[\left[\right]_\infty \right]_{\infty'} \Big]_1$;

$L_1 = a^{f(s_1)}$;

$L_2 = a^{2val(\gamma_1)}$;

$L_{j+1} = \phi,\ 2 \le j \le n$;

$L'_{j+1} = \phi,\ 1 \le j \le n$;

$L_\infty = L_{\infty'} = \phi$;

$$
\begin{aligned}
R_1 = \{&1 : c_1 a^{f(p)} \to c_1 a^{f(q)}(a^2, in_{t'}) \mid (p, \gamma_t^+, q) \in I\} \cup \\
&\{2 : c_1 a^{f(p)} \to c_1 a^{f^{(1)}(q)}(a^{b(t)}, in_{t'}), \\
&\ 3 : c_1 a^{f^{(1)}(q)} \to c_1 a^{f^{(2)}(q)}, \\
&\ 4 : c_1 a^{f^{(2)}(q)+b(t)+2} \to c_1 a^{f(q)}, \\
&\ 5 : c_1 a^{f(p)} \to c_1 a^{f^{(3)}(p)}, \\
&\ 6 : c_1 a^{f^{(3)}(p)} \to c_1 a^{f^{(1)}(q)}(b(t), in_{t'}) \mid (p, \gamma_t^-, q) \in I\} \cup \\
&\ 7 : \{c_1 a^{f(p)} \to c_1 a^{f^{(4)}(q)}(a^{b(t)}, in_{t'}), \\
&\ 8 : c_1 a^{f^{(4)}(q)} \to c_1 a^{f^{(5)}(q)}(a^{b(t)}, in_{t'}), \\
&\ 9 : c_1 a^{f^{(5)}(q)+b(t)+2} \to c_1 a^{f^{(6)}(q)}(b(t), in_{t'}), \\
&10 : c_1 a^{f^{(6)}(q)+b(t)+2} \to c_1 a^{f^{(7)}(q)}(b(t), in_{t'}), \\
&11 : c_1 a^{f^{(7)}(q)+b(t)+2} \to c_1 a^{f^{(8)}(q)}, \\
&12 : c_1 a^{f^{(8)}(q)+b(t)+2} \to c_1 a^{f(q)} \mid (p, \gamma_t^{=0}, q) \in I\} \cup \\
&\{13 : c_1 a^{f(s_f)} \to c_1(a^{f(s_f)}, out), 14 : c_1' a \to c_1'(a, in_{\infty'})\}; \\
R_{\infty'} = \{&15 : c_{\infty'} a \to c_{\infty'}(a, in_\infty)\}, \\
R_\infty = \{&16 : c_\infty a \to c_\infty(a, out)\}, \\
R_{j'} = \{&17 : c_{t'} a^2 \to c_{t'}(a^2, in_j), \\
&\ 18 : c_{t'} a^{b(t)+2} \to c_{t'}(a^{b(t)+2}, out), \\
&\ 19 : c_{t'}^{(1)} a^{b(t)} \to c_{t'}^{(1)}(a^{b(t)}, out), \\
&\ 20 : c_{t'}^{(1)} a^2 \to c_{t'}^{(1)}(a^2, in_j), \\
&\ 21 : c_{t'}^{(1)} a^{b(t)-2} \to c_{t'}^{(1)}(a^{b(t)-2}, out), \\
&\ 22 : c_{t'}^{(2)} a^2 \to c_{t'}^{(2)}(a^2, out)\} \\
R_j = \{&23 : c_t a^2 \to c_t(a^2, out)\},
\end{aligned}
$$

This proof requires the use of a unique-sum set U with at least $n + 9|S| + 1$ elements. Different multiplicities of a, where the multiplicities are elements in U, are associated with each of the registers and instructions of M. This is performed by the function $b : \{1, \ldots, n\} \to U$ and the nine functions $f, f^{(1)}, \ldots, f^{(8)}$ all from S to U, all injective and with disjoint values. The exact definition of these ten functions is irrelevant for the proof, the only thing that is essential is that the different elements of U are at least 3 units apart from each other. The reason for this requirement is explained in the following.

The simulation performed by the P system Π is strongly based on the use of a unique-sum set and on the property that for such sets none of the elements can be obtained as a linear combination of the remaining elements in the set. During the computation of Π, different occurrences of the symbol a are present in the skin compartment. Specific sequences of applied rules allow Π to simulate instructions of M. Other sequences of applied rules let Π to never halt.

If rule 14 is applied, then an infinite loop, given by the repeated application of rules 15 and 16 starts. If this happens, then Π never halts.

The compartments $2 \leq j, j' \leq n + 1$, in Π are associated in a unique way to the registers of M. If during a simulation, register γ_j contains the value $val(\gamma_j)$, then the sum of the occurrences of the symbol a present in j and j' is $2val(\gamma_j)$. The addition of 1 to register γ_j is then simulated with the addition of

two occurrences of a to compartment j'. The subtraction of 1 from register γ_j is simulated with the removal of two occurrences of a from compartment j'.

During the simulation, if compartment j' contains at least 2 occurrences of a, then either rule 17 or rule 20 can be applied. The passage of a^2 from compartment j to compartment j' is performed by the application of rule 23. This means that if both compartments j and j' contain at least 2 occurrences of a, then these occurrences keep been moved to j' and j, respectively. In the following description we avoid to repeat that the application of these rules can take place during the simulation of the instructions of M.

If the skin compartment has $f(s)$ occurrences of a, then Π simulates M being in state s.

The simulation of instructions of the kind (p, γ_t^+, q) is performed by the sequential application of rule 1 and then either rule 17 or 20. In this way the number of occurrences of a in the skin compartment changes from $f(p)$ to $f(q)$, indicating that Π simulates the state change of M from p to q, and 2 occurrences of a pass to compartment t' and then to compartment t, simulating the addition of 1 unit to register γ_t.

To describe the simulation of instructions of the kind (p, γ_t^-, q) we have to consider different configurations of t and t' when rule 2 is applied:

i) a^2 is present in t' and $a^k, k \geq 2, k$ even, is present in t;
ii) a^2 is present in t' and t does not contain any a;
iii) t' does not contain any a and t contains a^2.

Because of rules 17, 20 and 23, it cannot be that when t' does not contain any a, t contains more than 2 occurrences of a.

In the following rule numbers in parenthesis denote that the rules are applied in parallel. The simulation of such instructions can non-deterministically start with either rule 2 or rule 5. If it starts with rule 2 and Π is in configuration *ii*, then the applied rules are (2, 17) (or (2, 18)) followed by (3, 19, 23) (or (3, 17, 20, 22, 23), or (3, 17, 21, 23)). In all these cases, rule 4 cannot be subsequently applied and Π starts an infinite loop. Similarly, if the simulation starts with rule 5 and Π is in configuration *iii*. In the other cases the simulation of the instruction can be completed.

Now we describe these possibilities one by one.

If Π is in configuration *i*, then (2, 17, 23) can be applied. When this happens compartment t' has $b(t)+2$ occurrences of a. If now rules (3, 18, 23) are applied, then in the following configuration rules (4, 17), and eventually also 23, can be applied. This is a simulation of (p, γ_t^-, q).

When $a^{b(t)+2}$ is present in t', then other groups of rules can be applied in parallel. It is important to notice that maximal strategy forces all these $b(t)+2$ occurrences of a to be subject to a rule. For instance, it can be that (3, 17, 19, 23) are applied. If this occurs, then, as rule 3 cannot be applied, rule 14 is applied starting in this way an infinite loop.

If Π is in configuration *iii*, then (2, 23) can be applied. What can happen next is similar to what just described.

If Π is in configuration *ii*, then rules (2, 17) are applied. When this happens $b(t)$ occurrences of a are present in t' and the applied rules let Π start an infinite loop.

If Π is in configuration *i*, then (5, 17, 23), (6, 17, 23) can be applied. When this happens compartment t' has $b(t) + 2$ occurrences of a and the simulation can go on as described in the above.

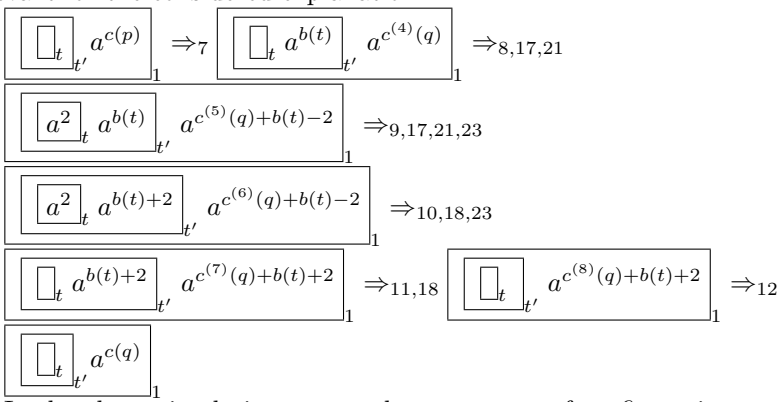
If Π is in configuration *ii*, then rules (5, 17), (6, 23) can be applied, again $b(t) + 2$ occurrences of a and the simulation can go on as described in the above.

If Π is in configuration *iii*, then rules (5, 23) can be applied. When this happens $b(t)$ occurrences of a are present in t' and the applied rules let Π start an infinite loop.

The simulation of instructions of the kind $(p, \gamma_t^{=0}, q)$ can start while t and t' are in one of the following configurations:

iv) a^2 is present in t' and at least a^2 is present in t;
v) a^2 is present in t' and t does not contain any a;
vi) t' does not contain any a and t contains a^2;
vii) both t and t' do not contain any a.

Only configuration *iv* can let Π not to end up in an infinite loop. We describe this simulation with a detailed description using a graphical representation of the compartments in Π. Here we indicate only the compartments and symbols relevant for the considered explanation.



In the above simulation many other sequences of configurations could occur but none of them would lead the system to a halt. For instance, instead of applying rules (8, 17, 21) the rules (8, 19) or (8, 17, 20, 22) could be applied but in both cases the computation would go on forever.

The reason why we ask that the elements of U are at least 3 units apart from each other is that because some configurations see 'spare' a's present in the skin compartment. For instance, in the above when rule 1 is applied the skin compartment contains $a^{c(q)+2}$. If the elements of U were not at least 3 units apart from each other there could be a rule using the number of occurrences of a's together with other a's present in the skin compartment. The application of this rule could let Π not to simulate M. As we ask the elements of U to be at least 3 units apart from each other, then this cannot happen.

When only $f(s_f)$ occurrences of a are present in the skin compartment, then the application of rule 13 halts the computation.

The simulation of M is faithful, that is Π cannot simulate sequences of instructions that cannot be performed by M. This means that if Π starts with a configuration in which $a^{f(s_1)}$ is present in the skin compartment and $a^{2val(\gamma_1)}$ is present in compartment 2, then Π halts with no a's in any of its compartments if and only if M would reach the final state with all registers empty if it started from the configuration $(s_1, val(\gamma_1), 0, \ldots, 0)$.

It is important to notice that the presence of catalysts is essential. If they were not present, then, for instance, rules 17 and 20 would be equal. Moreover, the fact that different rules in the same compartment share the same catalyst (as, for instance, rules 20 and 21) does not allow these rules to be used at the same time. It is also fair to say that not all catalysts used in this proof are needed. For instance, the ones in the $j, 2 \leq j \leq n$, compartments can be removed without any effect on the result (if one keeps maximal strategy as operational mode). As our aim was not to minimise the number of rules using catalysts, we let all rules to have catalysts. □

Considering Theorem 5 and Proposition 1, we have:

Corollary 3.

$\mathbb{N}_1\mathsf{aO}_1\mathsf{P}_9(pmcat_{15})_{ms} = \mathbb{N}_1\mathsf{RE}$;

$\mathbb{N}\mathsf{aO}_1\mathsf{P}_{9,-c}(pmcat_{15})_{ms} = \mathbb{N}\,\mathsf{RE}$;

$\mathbb{N}_1\mathsf{aO}_1\mathsf{P}_7(pmcat_{11})_{ms} = \mathbb{N}_1\mathsf{RE}$ *if a specific input format is used;*

$\mathbb{N}\mathsf{aO}_1\mathsf{P}_{7,-c}(pmcat_{11})_{ms} = \mathbb{N}\,\mathsf{RE}$ *if a specific input format is used;*

$\mathbb{N}\mathsf{aO}_1\mathsf{P}_{5,-c}(pmcat_7)_{ms} \supseteq \mathbb{N}\,\mathsf{REG}$.

5 Final remarks

Theorem 3 partially answers suggestion for research 5.3 in [6] and a problem in [11].

The suggestion for research in [6] proposed to understand why maximal parallelism was needed for a model of P systems to have a computational power equivalent to the one of partially blind register machines. Here we proved that increasing the number of compartments, these devices can simulate program machines.

The problem in [11] asked to investigate weather P systems with symport/ antiport using only one symbol and operating under maximal parallelism are universal. Theorem 3 proves that when these devices operate under maximal strategy they are indeed universal. We were not able to prove if the same holds when maximal parallelism is in place.

It is worth saying that studies of different operational modes are present in [6, 4].

We also did not succeed in determining the power of the P systems considered in Theorems 3 and Theorem 5 when they operate in a sequential (asynchronous) mode.

Acknowledgements

We thankfully acknowledge the feedback Oscar Ibarra and one anonymous referee provided to previous versions of the present paper.

References

1. A. Alhazov and R. Freund. P systems with one membrane and symport/antiport rules of five symbols are computationally complete. In M. A. Gutiérrez-Naranjo, A. Riscos-Núñez, F. R. Romero-Campero, and D. Sburlan, editors, *Proceedings of the Third Brainstorming week on Membrane Computing. Sevilla, Spain, January 31 - February 4, 2005*, pages 19–28. Fénix Editoria, Sevilla, 2005. Available from [20].
2. A. Alhazov, R. Freund, and M. Oswald. Symbol/membrane complexity of P systems with symport/antiport. In R. Freund, G. Lojka, M. Oswald, and G. Păun, editors, *Membrane Computing. 6^{th} International Workshop, WMC 2005, Vienna, Austria, July 18-21, 2005, Revised Selected and Invited Papers*, volume 3850 of *LNCS*, pages 96–113. Springer-Verlag, 2005.
3. Ja. M. Barzin'. On a certain class of Turing machines (Minsky machines). *Algebra i Logika*, 1(6):42–51, 1962/1963. (in Russian) MR 27 #2415.
4. R. Freund and S. Verlan. A formal framework for static (tissue) P systems. In G. Eleftherakis, P. Kefalas, G. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing. 8^{th} International Workshop, WMC 2007, Thessaloniki, Greece, June 2007, Revised Selected and Invited Papers*, volume 4860 of *LNCS*, pages 271–284. Springer-Verlag, 2007.
5. P. Frisco. On s-sum vectors. Technical report, Heriot-Watt University, 2008. HW-MACS-TR-0058, available at `http://www.macs.hw.ac.uk:8080/techreps/build_table.jsp`.
6. P. Frisco. *Computing with Cells. Advances in Membrane Computing.* Oxford University Press, 2009.
7. P. Frisco. Conformon P systems and topology of information flow. In G. Păun, editor, *Membrane Computing. 10^{th} International Workshop, WMC 2009, Curtea de Arges, Romania, August 24-27, 2009, Revised Selected and Invited Papers*, volume 5957 of *LNCS*, pages 30–53. Springer-Verlag, 2009.
8. S. A. Greibach. Remarks on blind and partially blind one-way multicounter machines. *Theoretical Computer Science*, 7:311–324, 1978.
9. J. E. Hopcroft and D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.
10. O. H. Ibarra. On membrane hierarchy in P systems. *Theoretical Computer Science*, 334:115–129, 2005.

11. O. H. Ibarra and S. Woodworth. On symport/antiport P systems with small number of objects. *International Journal of Computer Mathematics*, 83(7):613–629, 2006.
12. O. H. Ibarra, H.-C. Yen, and Z. Dang. On various notions of parallelism in P systems. *International Journal of Foundations of Computer Science*, 16(4):683–705, 2005.
13. M. L. Minsky. *Computation: Finite and Infinite Machines*. Automatic computation. Prentice-Hall, 1967.
14. A. Păun and G. Păun. The power of communication: P systems with symport/antiport. *New Generation Computing*, 20(3):295–306, 2002.
15. G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 1(61):108–143, 2000.
16. G. Păun. *Membrane Computing. An Introduction*. Springer-Verlag, 2002.
17. G. Păun, J. Pazos, M. J. Pérez-Jiménez, and A. Rodriguez-Paton. Symport/antiport P systems with three objects are universal. *Fundamenta Informaticae*, 64:1–4, 2005.
18. G. Păun, G. Rozenberg, and A. Salomaa, editors. *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.
19. Z. Qi, J. You, and H. Mao. P systems and petri nets. In C. Martín-Vide, G. Mauri, G. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing. International Workshop, WMC 2003, Tarragona, Spain, July 17-22, 2003, Revised Papers*, volume 2933 of *LNCS*, pages 286–303. Springer-Verlag, 2004.
20. The P Systems Webpage. `http://ppage.psystems.eu/`.